Introduzione agli algoritmi
Appello Straordinario del 12/4/2019
T. Calamoneri - E. Fachini - R. Petreschi

Esercizio 1.

Si consideri l'array *A = [19, 18, 10, 15, 20, 9, 5, 13, 2, 4, 15]* e si verifichi se soddisfa la proprietà di essere un max-heap.

Se non lo fosse, lo si renda un max-heap, dettagliando tutti i passaggi.

Si illustri sul max-heap l'algoritmo per l'estrazione del massimo, mettendo in evidenza i cambiamenti dell'array *A* per ogni passo di esecuzione dell'algoritmo; di tale algoritmo si fornisca poi lo pseudocodice ed il costo computazionale.

Esercizio 2.

Si consideri il problema di ordinare un array *A*; un possibile algoritmo può essere schematizzato come segue:

- a partire dai dati in *A*, costruisci un albero binario di ricerca;
- esegui una visita in in-ordine dell'albero binario di ricerca.

Si fornisca il tempo di esecuzione asintotico di questo algoritmo nel caso peggiore e si discuta come esso cambierebbe se l'albero binario di ricerca fosse bilanciato (AVL o Rosso-nero).

Esercizio 3.

Si considerino i seguenti algoritmi che prendono entrambi in input un albero binario di ricerca con *n* chiavi. Per ogni algoritmo si spieghi brevemente cosa fa, e si fornisca poi il tempo di esecuzione asintontico nel caso migliore e peggiore, giustificando la risposta.

Algoritmo1(*T , k***)**
input: *T* è un albero binario di ricerca con chiavi intere e *k* è un intero
```
if T==nil return false
if T.key==k return true
if Algoritmo1(T.left) return true
else return Algoritmo1(T.right)
```

Algoritmo2(*T, k1, k2***)**
input: *T* è un albero binario di ricerca con chiavi intere e *k1* e *k2* sono interi
```
if T==nil return 0
if k1 > k2 scambia i valori di k1 e k2
r=0
if T.key < k2
    r = r + Algoritmo2(T.right,k1,k2)
if T.key > k1
    r = r + Algoritmo2(T.left,k1,k2)
if T.key < k2 and T.key >k1
    r = r+1
return r
```